GIGA+
Scalable file system directories

Swapnil Patil and Garth Gibson
{firstname.lastname} @ cs.cmu.edu
Carnegie Mellon University

HECFSIO Workshop 2010

Demand for massive directories

- New applications that use file systems as a fast, lightweight "database"
  - All clients creating large number of files in a single directory at high speeds[hpcs08]
  - Examples: N-to-N checkpoints, science apps
- Highly concurrent execution of data-intensive apps
  - By 2020, Exascale-era clusters expected to have up to one billion cores[darpa08-study]
  - Even simple workloads can stress the metadata service

State of file systems today

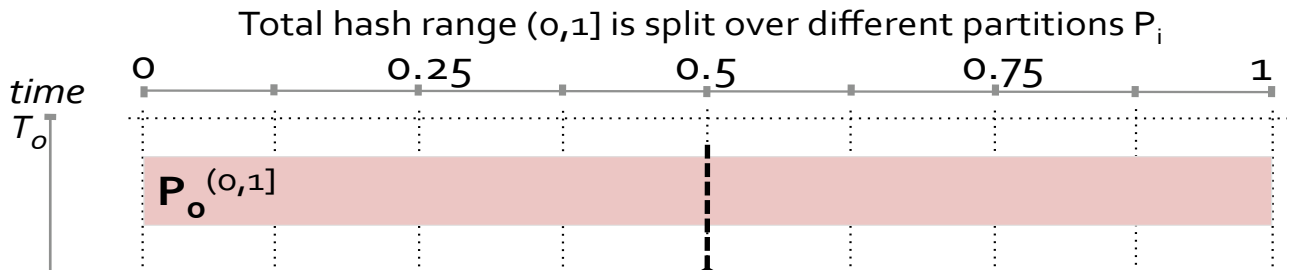
- Provide high parallelism on the data path, not much on the metadata path
  - Either store directories on one metadata server
  - Or avoid many small files using new semantics[GoogleFS]
  - Distributed directories: GPFS[Schmuck02] and Lustre[Lustre]

- Goal: Highly parallel directory indexing (GIGA+)
  - Push scalability limits while maintaining UNIX FS and POSIX-like semantics

Challenges in distributed indices

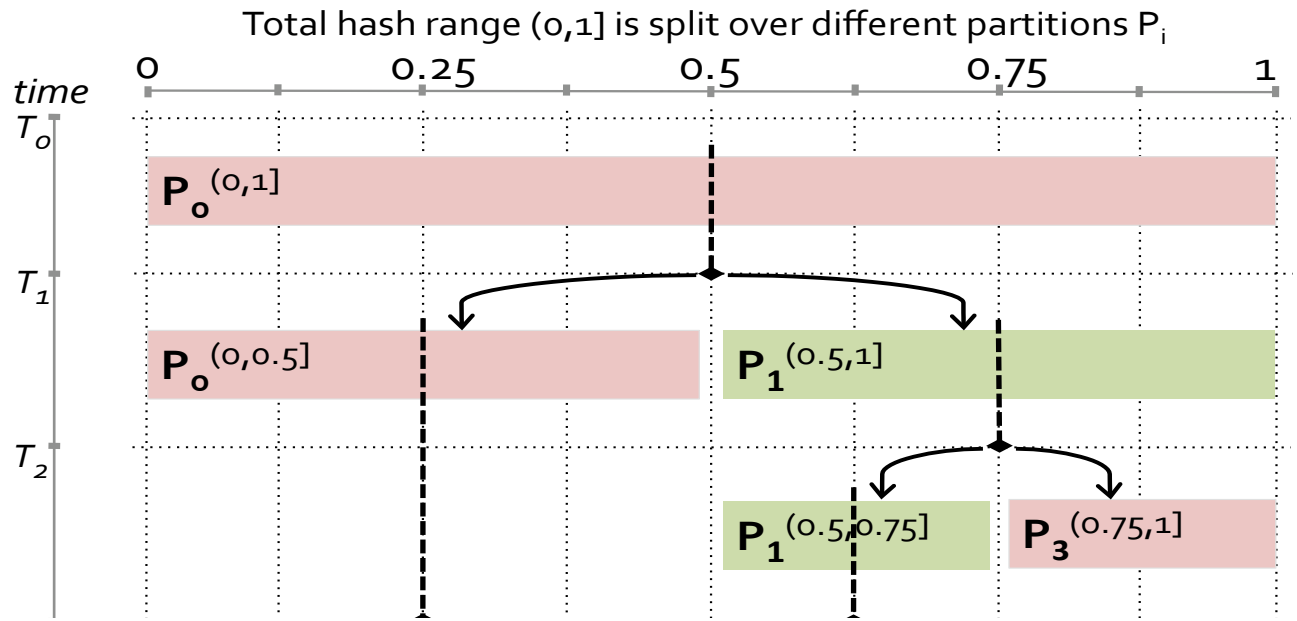
- *Setup*: Directory partitioned, mapped to servers
- How do servers expand partitions?
  - Serialized order of splitting[Litwin96] or synchronized splitting using locks[Schmuck02]
  - GIGA+ avoids both serialization and synchronization
- How do clients learn about new partitions?
  - Servers ensure client's mapping consistency[Schmuck02]
  - GIGA+ tolerates inconsistent mapping state at clients
    - Note: Apps see strong *data* consistency

GIGA+ illustration



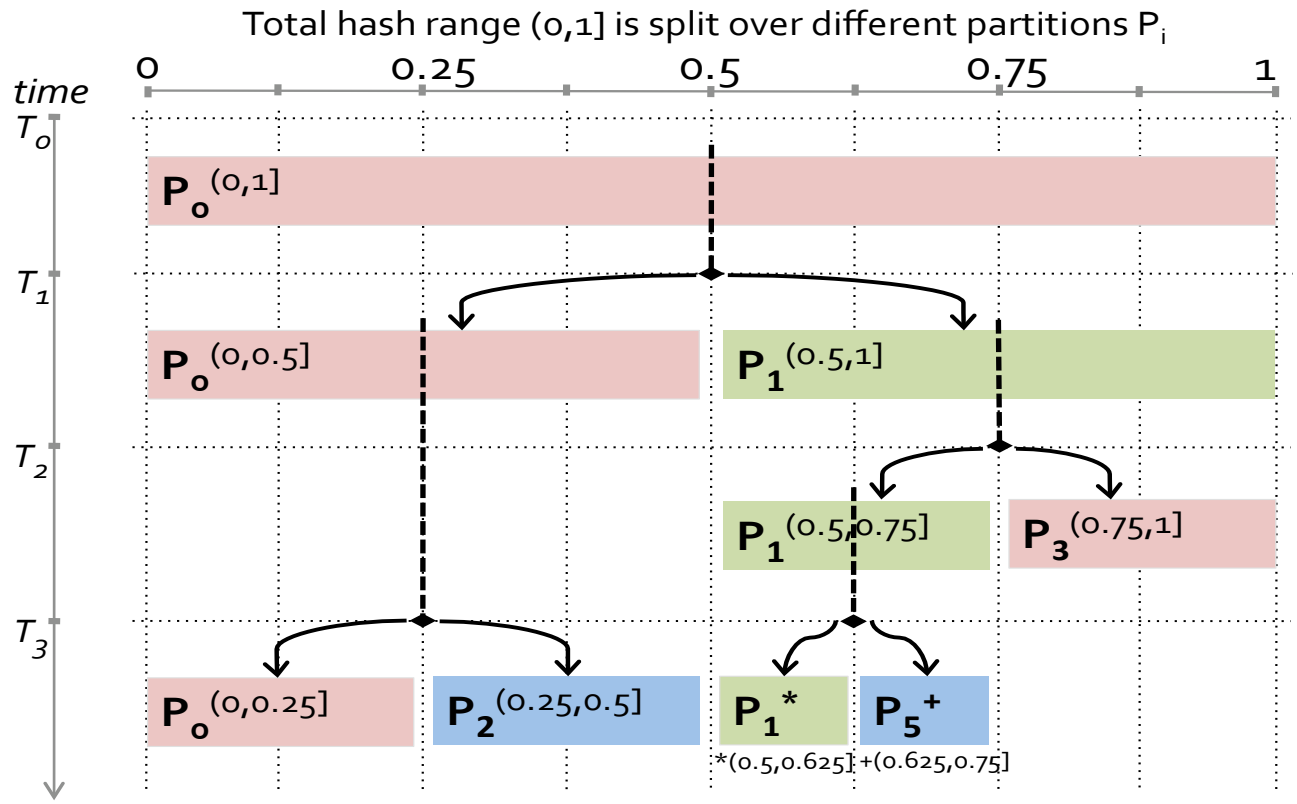
- Hash-based partitioning
- Design decision: incremental growth
  - Keeps small directories on a single server
  - 99.9% of directories less that 8K entries[Dayal08]

GIGA+ illustration (contd.)



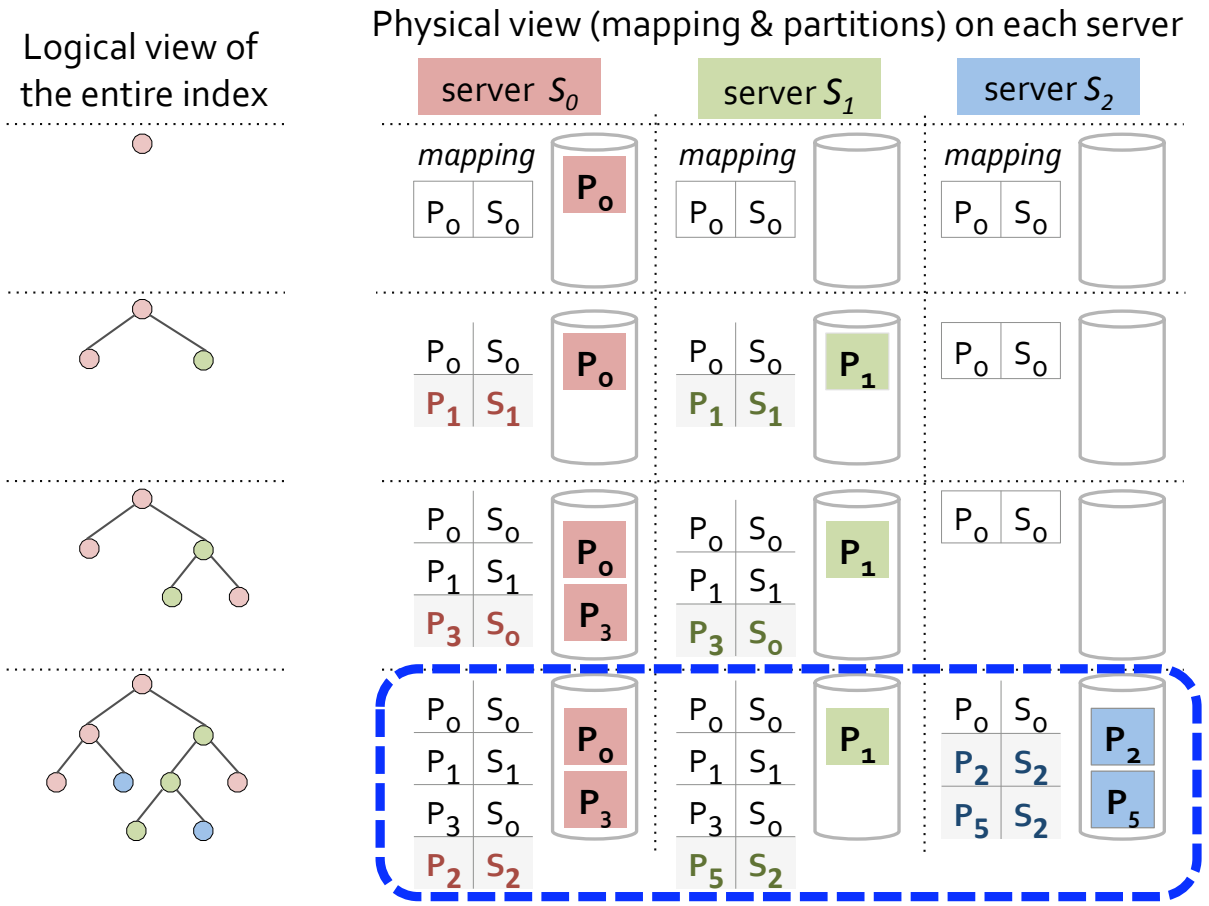
- Repeated splitting, proportional to the size
  - Avoids one-time full width splitting; may lead to many small-sized partitions

GIGA+ illustration (contd.)



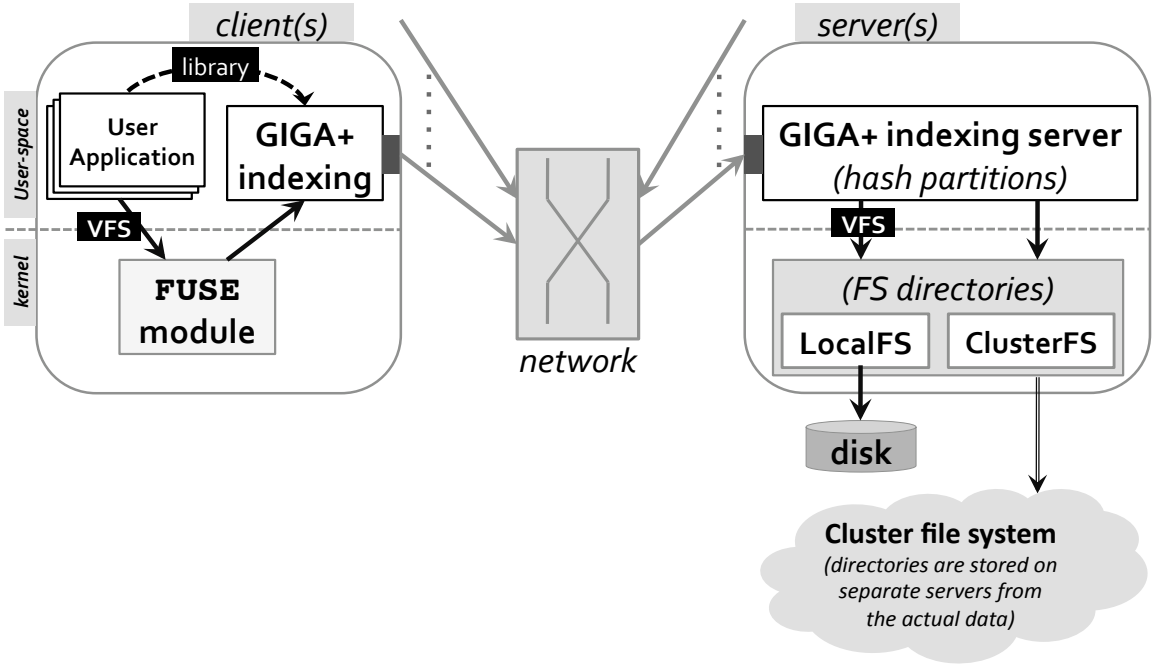
- Servers split independently, without any synchronization
  - Split only until all servers have appropriate # of partitions

GIGA+ illustration (contd.)



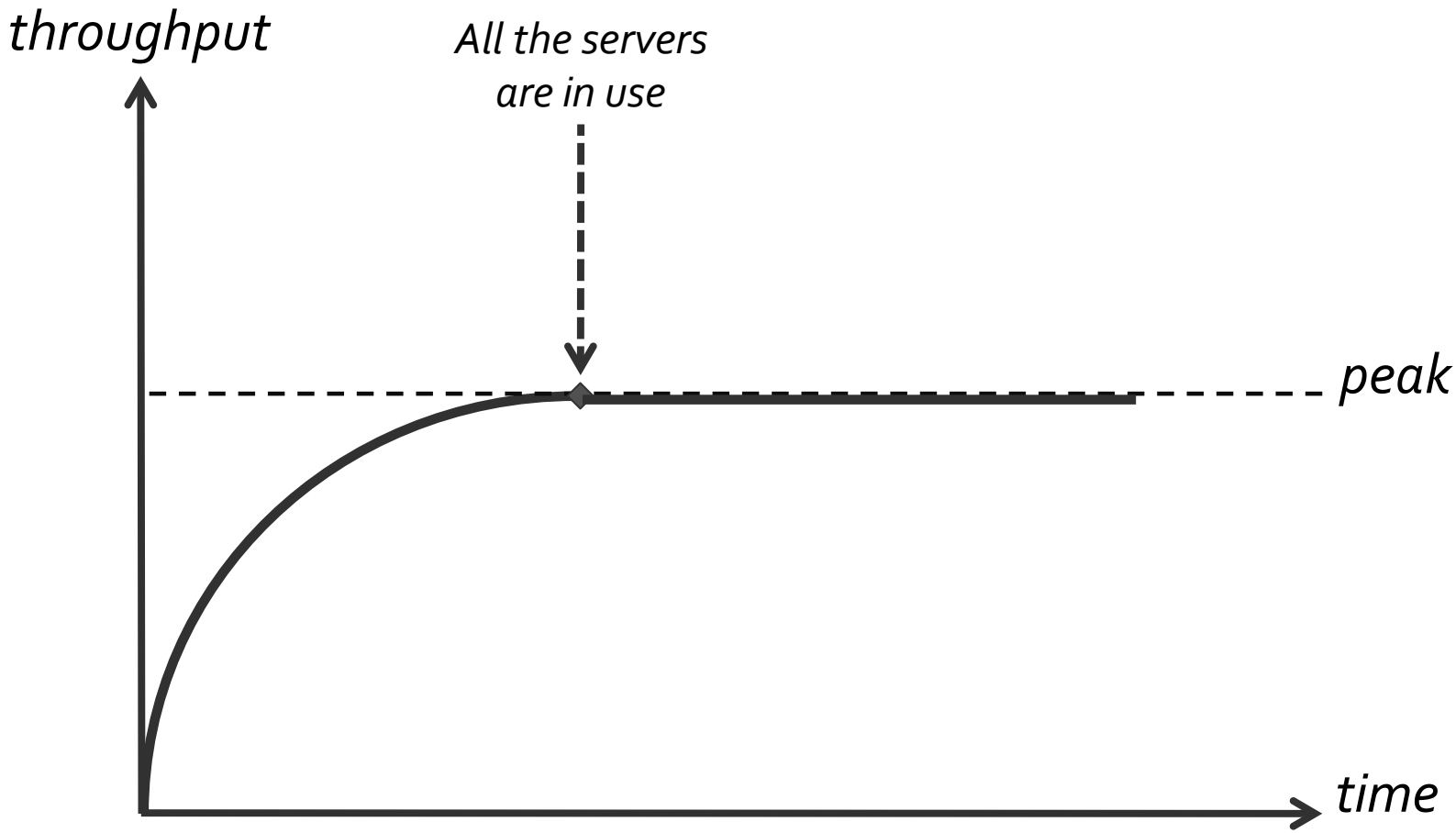
- Servers keep partial mapping view of the index
  - View includes a partition and a "history" of its splits

Experimental evaluation



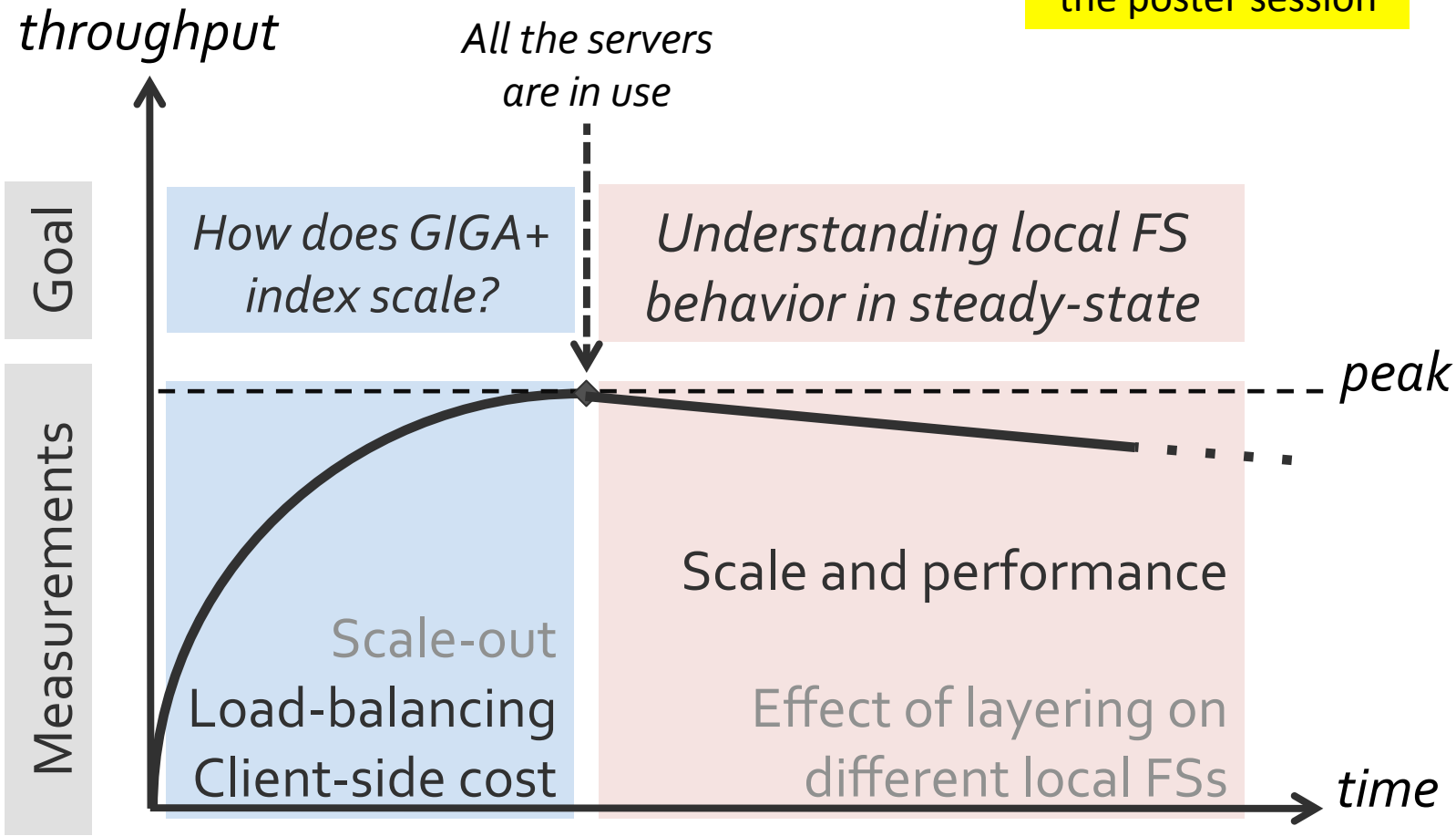
- Benchmark: Modified `mdtest` (http://sourceforge.net/projects/mdtest/)
  - Concurrent create workload that creates files proportional to the # of servers (400K file on 1 server, 800K on 2, and so on …)
- Setup: 64 nodes, dual quad-cores with 16GB RAM with a 10 GigE network
  - Each machine has SATA disks running a local file system
  - 8 client threads generating work per server

Experimental evaluation

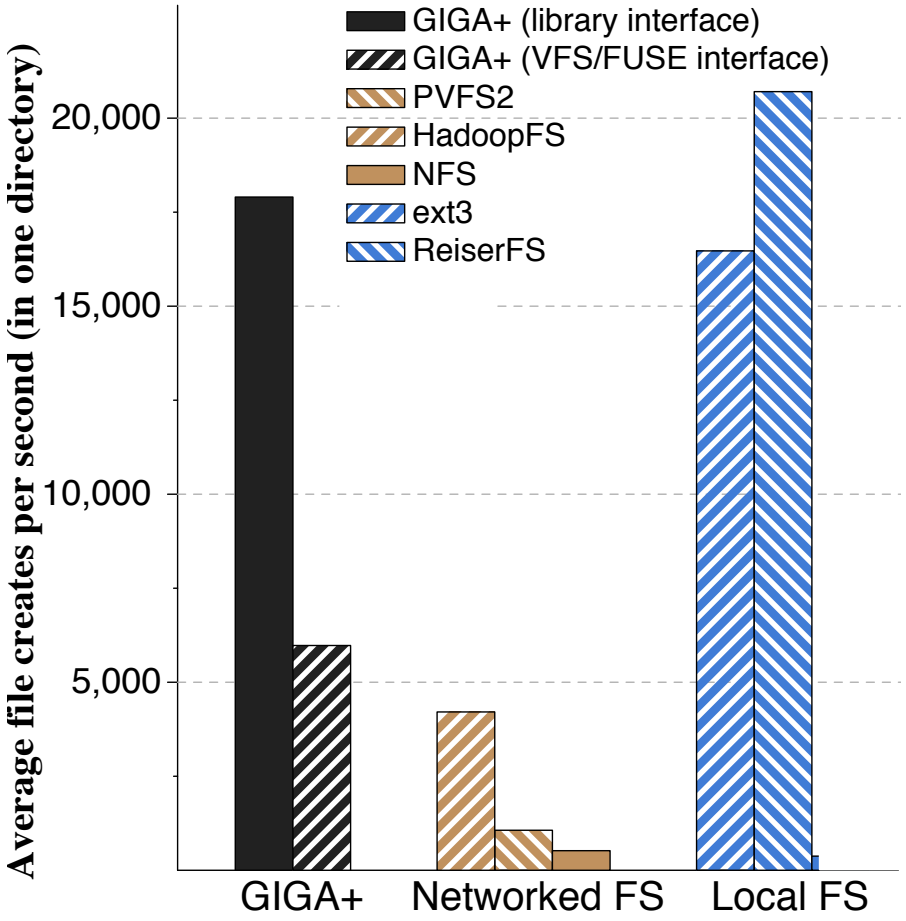


Experimental evaluation

Full evaluation @ the poster session

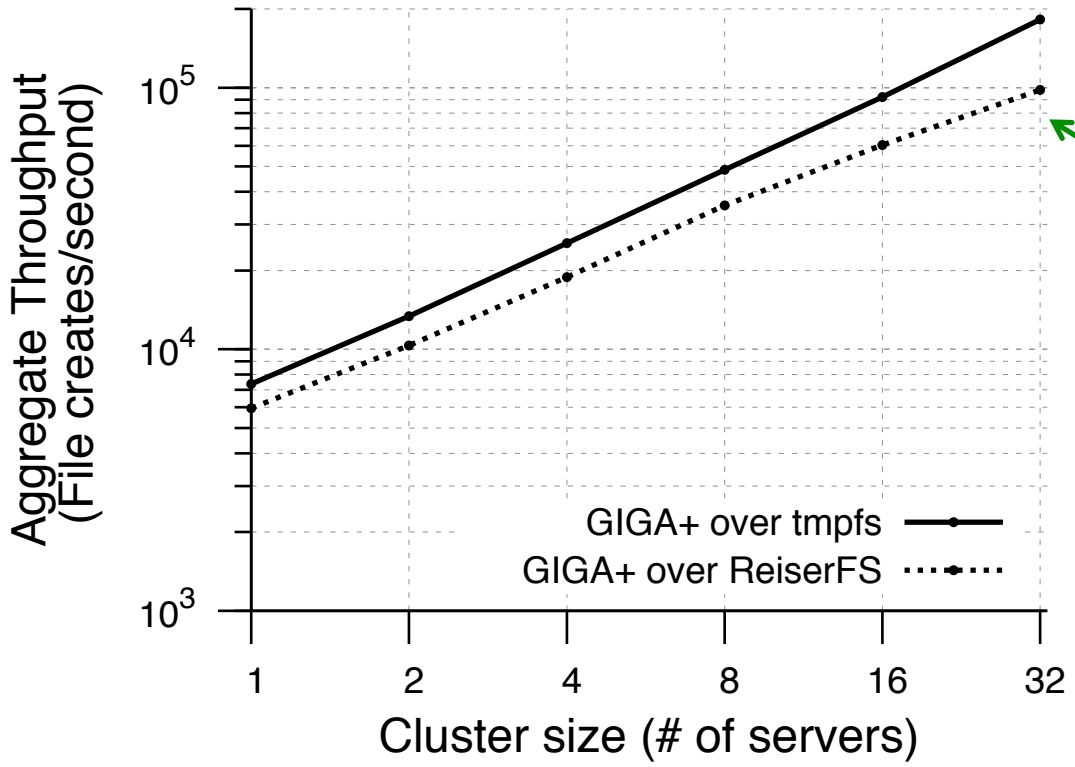


Single-node baseline performance



- VFS/FUSE sends three RPC requests for every file create
- On par with the real distributed FSs
- Local FS configurations
  - Client threads create files in a local directory

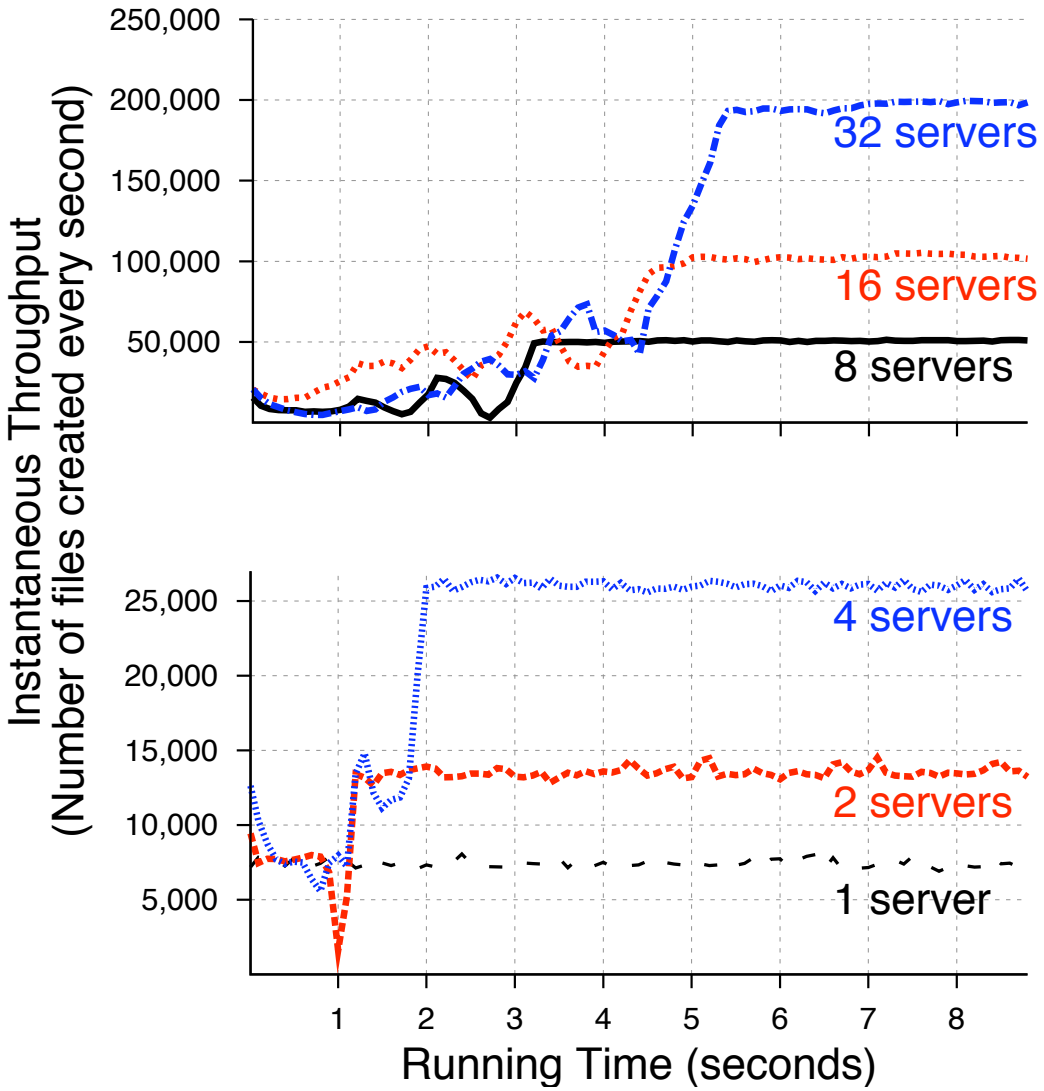
Scaling FS directories



~98K creates/second

Sub-linear scaling due to the implementation decisions of localFS (ReiserFS on disk)

Incremental scale-out performance

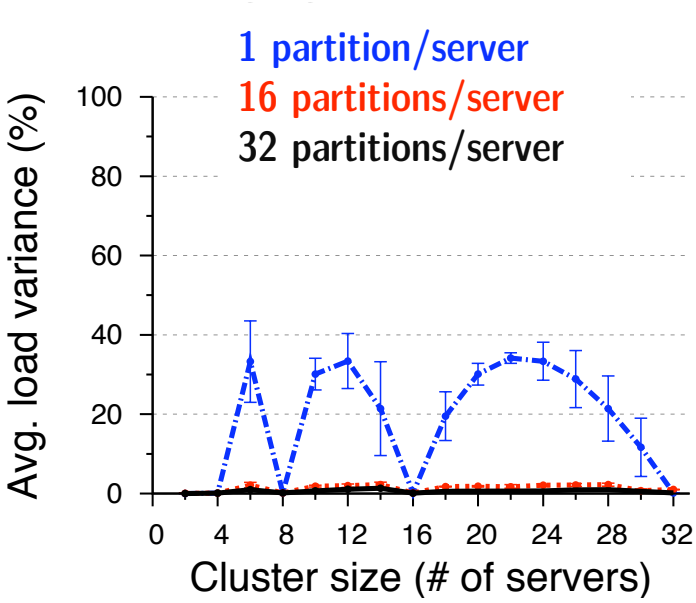


- Linear scaling after distributing over all servers
- Throughput drops during incremental splitting

Load-balancing effectiveness

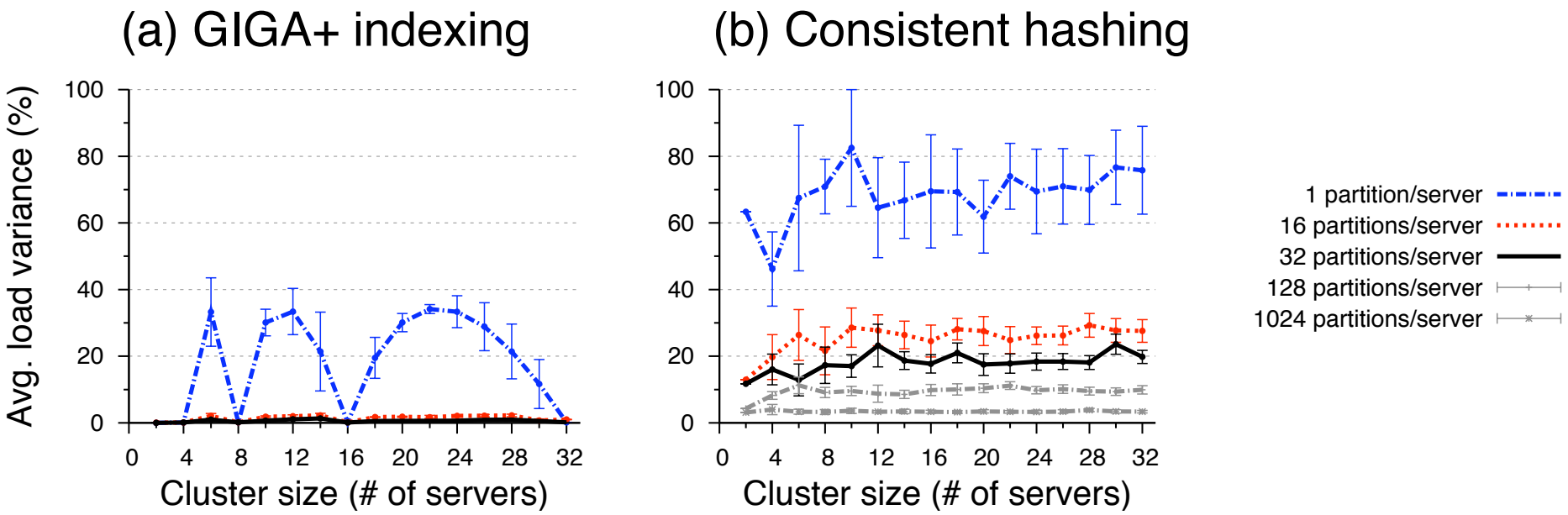
- Find load variance for a server
  - Average (95% CI) over all servers

$$\frac{|L_i - L_{avg}|}{L_{avg}}$$



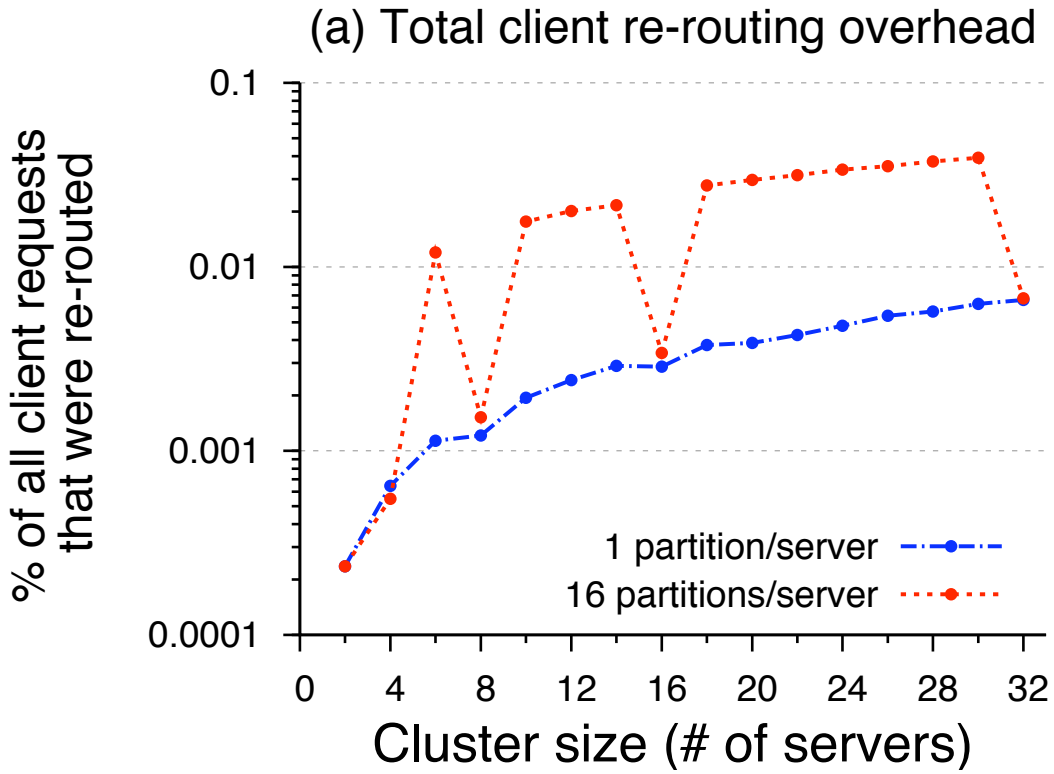
- Load balanced for *pow(2)*
- For other cases more partitions per server reduces variance

Load-balancing effectiveness



- Needs two orders of magnitude less partitions on each server than consistent hashing

Low cost of weak mapping at clients



- **Negligible rerouting overhead at the clients**

Summary of GIGA+

- Push the limits of scalability for FS directories
  - store billions of files per directory over hundreds of servers
  - sustain 100,000s of mutations/second

- Exploit opportunities to parallelize indexing
  - Eliminate system-wide synchronization and serialization
  - Avoid strong consistency (for everything other than data)
- Maintain UNIX FS and POSIX-like semantics
  - Complement existing cluster FSs and run unmodified apps